New Generation Computing, 20(2002)307-315
Ohmsha, Ltd. and Springer-Verlag

Experimental Efficiency of Programmed Mutagenesis

Julia KHODOR and David K. GIFFORD
Massachusetts Institute of Technology
200 Technology Square, Cambridge,
MA 02139, USA
jkhodor@mit.edu, gifford@mit.edu

Received 5 October 2001
Revised manuscript received 7 March 2002

Abstract Mismatched DNA annealing followed by strand replication can cause the programmed evolution of DNA sequences. We have reported that this process is theoretically equivalent in computational power to a desktop computer by demonstrating a constructive way to encode arbitrary computations as DNA molecules within the framework of programmed mutagenesis, a system that consists solely of cycles of DNA annealing, polymerization, and ligation.[1,2] Thus, programmed mutagenesis is theoretically universal and we report here the experimental efficiency of its primitive operations. The measured efficiency of an in vitro programmed mutagenesis system suggests that segregating the products of DNA replication into separate compartments would be an efficient way to implement molecular computation. For computer science, using single DNA molecules to represent the state of a computation holds the promise of a new paradigm of composable molecular computing. For biology, the demonstration that DNA sequences could guide their own evolution under computational rules may have implications as we begin to unravel the mysteries of genome encoding and natural evolution.

Keywords: Programmed Mutagenesis, Universal System, Biological Computing, String Rewrite Systems, Turing Machines.

§1 Introduction

Existing work on DNA computing can be categorized as practical systems with limited computational power, theoretical systems that are not presently realizable, or practical systems that are universal, but not composable. A generate-and-test style approach was used to solve a directed Hamiltonian path problem[3] and was generalized to other problems in NP,[4] but is not universal. Circuit simulations[5] may be universal, but are not composable, unless a unique DNA

sequence is used for each bit position. Autonomous string systems based on hairpin formation[6,7] posses interesting computational behaviors, but are neither composable, nor universal. Insertion/deletion systems[8,9] are theoretically universal, but enzymes with required activities are not presently known. Both splicing systems and sequential mutagenesis of DNA that require sequence specific separations are theoretically universal[10~12] but have not been demonstrated to be realizable. DNA tiling systems are universal,[13] and certain proposed implementations can be composable, but the one implemented to date[14] is not. We will call a computing system composable when a first computation results in a single molecule that can be used directly by a second computation as input without modification.

Programmed mutagenesis is a nucleic acid computational system that is both universal and composable and for which some experimental progress has been demonstrated. Programmed mutagenesis[16] is an in-vitro mutagenesis technique based on oligonucleotide-directed mutagenesis[17] which produces sequence specific rewriting of DNA molecules. Like oligonucleotide-directed mutagenesis, programmed mutagenesis does not mutate existing strands of DNA, but instead uses DNA polymerase and DNA ligase to create copies of template molecules, where the copies have engineered mutations at sequence specific locations. Every time a programmed mutagenesis reaction is thermal cycled a rewriting event occurs. Because the technique relies on sequence specific rewriting, multiple rules can be present in a reaction at once, with only certain rules being active in a given rewriting cycle. Furthermore, the ability for the system to accommodate inactive rules allows it to proceed without human intervention between cycles. Programmed mutagenesis systems are composable because the output from one computation can be directly used as input to a second computation.

§2 Unary Counter

An example programmed mutagenesis system that implements a unary counter is shown in Fig. 1. The template (I) contains an encoding of a series of symbols XZZZZZ embedded in a noncoding region. The machine is called a unary counter because we can think of the counter as being incremented every time the system is thermocycled. We say that the number of symbols other than Z (i.e. X and Y) in the coding region minus one is the current count in the counter. Thus, template I carries the count of zero, since it contains one symbol other than Z. Every mutagenic cycle rewrites another Z into either X or Y, incrementing the counter by one.

The entire region is cloned into a plasmid using Eco RI and Hind III restriction enzymes. The outer primer MLP is part of the noncoding region and the outer primer MRP is a part of the plasmid sequence. Each symbol used in the system (X, Y, and Z) is encoded by a 12-nucleotide long sequence of DNA. The actual encodings used for these symbols are shown in Fig. 2. The bases at which encodings are mismatched are indicated. X and Y both differ from Z by two mismatches and from each other by 4 mismatches. The system was designed such that any oligonucleotide binding with two or fewer mismatches would be able

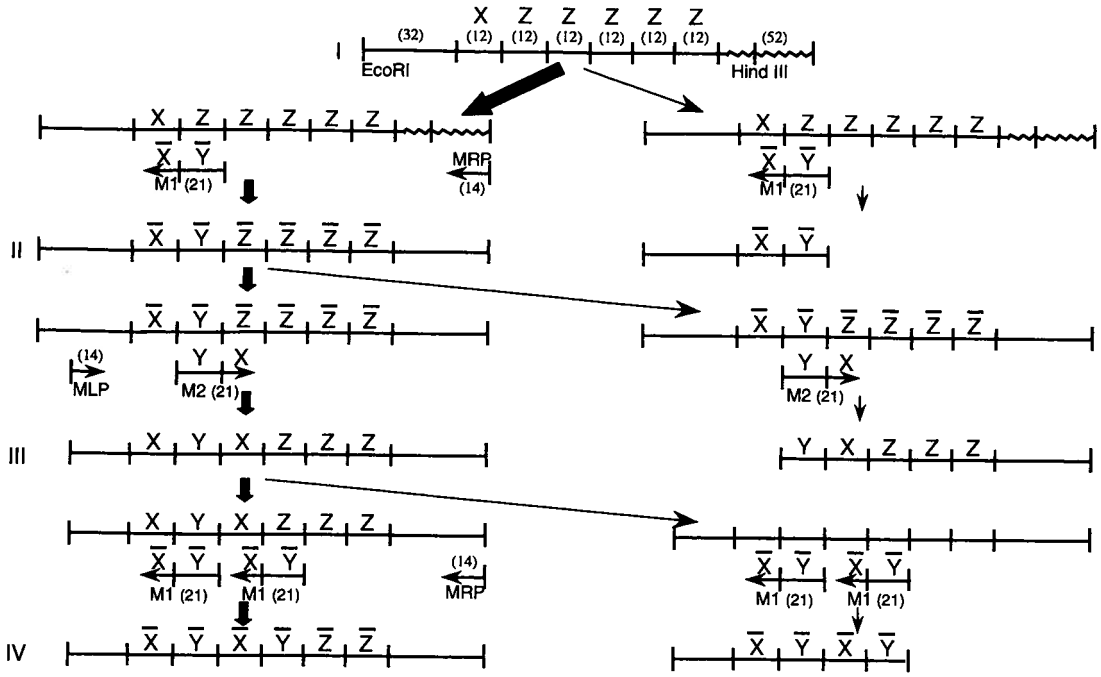

Fig. 1 Schematic representation of the unary counter. M1 and M2 are mutagenic rule oligonucleotides; MRP and MLP are perfectly matched outside oligonucleotides. Note that a rule incorporated in the previous cycle becomes part of the template for the following cycle. Bold arrows denote the transitions which carry the computation forward. Also shown are the events which lead to creation of the characteristic bands for each cycle. These characteristic bands are the results of failed ligation events, so named because they represent the result of a failed ligation of the successful extension of the perfectly matched outside primer to the successful extension of the mutagenic rule oligonucleotide.

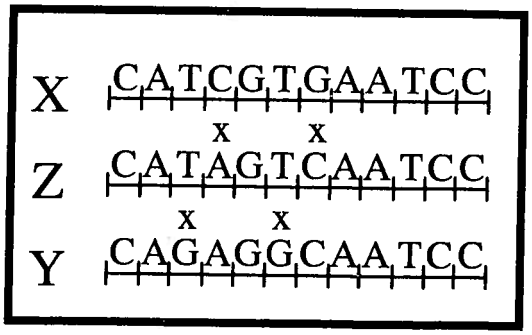

Fig. 2 Encodings of the symbols used in implementing the unary counter machine in Fig. 1. All sequences are given in the 5′-to-3′ orientation, and mismatch locations are indicated.

to bind, extend, and be ligated to, and any oligonucleotide binding with more than two mismatches would not be able to interact with the template. Mismatch locations were designed to minimize the opportunity for inappropriate binding and to negate the ability of an oligonucleotide bound with four mismatches to be ligated to. This was possible because of the requirements on the alignment of the oligonucleotides introduced by enzymes used in this experimental system.

The DNA sequences for the system we are testing have been selected using the SCAN program[15] to search a large sequence space constrained by chosen mismatch geometry. SCAN chooses sequences that have optimum annealing properties, lack harmful secondary structure, and do not form primer dimers. We chose fairly strict thermodynamic constraints in order to prevent inappropriate binding of the rule oligonucleotides, as well as any undesirable interaction between primers. Nevertheless, the search space remained too large, and needed to be further constrained. As mentioned above, we chose to constrain that space by defining the geometry of mismatches for the rewrite rules.

Each oligonucleotide able to anneal in the system is expected to be extended by the polymerase to the end of the available template or until a product of another extension is encountered. When one strand is extended to encounter another oligonucleotide positioned on the template, a ligation event is expected to occur. Ligation is not 100% efficient, and results of failed ligations are expected and are termed characteristic bands of a particular cycle. Such characteristic bands, as well as full-length products are illustrated in Fig. 1.

As discussed above, failed ligations result in characteristic products. We designed the system such that all these characteristic products have unique and easily distinguishable lengths. We use the appearance of these unique length products to judge whether a cycle of mutagenesis has indeed taken place. Because all these products have unique lengths, and the appearance of characteristic products of cycle n always precedes the appearance of characteristic products of cycle $n+1$, we did not feel it was necessary to clone and sequence these products to definitively verify their identity. A more complete study would perhaps undertake this cloning effort in addition to the characteristic length verification.

Mutagenic oligonucleotide M1 participates in creating a first cycle product (II) that contains a different sequence than the first cycle template (I). This change permits mutagenic oligonucleotide M2 to bind to product II in cycle two, producing another new product III that incorporates M2. Product III contains a sequence that permits oligonucleotide M1 to bind in a yet another location in the third cycle yielding product IV.

Thus a sequence of related novel products (II $\rightarrow$ III $\rightarrow$ IV) is created in a specific order. Outer primers and ligation are used to create full-length products, and all of the enzymes used in the system are thermostable which allows the system to be thermal cycled for progress. We recently answer the question of the computational power of such a system, by showing that it is theoretically universal.[1] Here we examine the practical feasibility of the underlying specific annealing, polymerization, and ligation operations.

Programmed mutagenesis relies on mismatches in rewrite rules to se-

quence program steps. Our abstract model of programmed mutagenesis uses the number of mismatches as a sole determining factor of the ability of a primer (rewrite rule) to bind, extend, or ligate to related DNA sequences. We do not model all of the secondary factors that influence these processes in part because there is not enough information to construct a reliable model, but primarily because mathematical insight would be impossible in an overly detailed model.

In programmed mutagenesis, we represent states and symbols by nucleotide sequences and enact state transitions by primer extension reactions. Note that rules become part of the template for the next cycle. Thus, the programmed mutagenesis rules do not take the form of antecedent $\rightarrow$ consequent, but rather are consequents searching for any antecedent that is within a certain number of mismatches.

The challenge in creating an encoding for any programmed mutagenesis system lies in the need to find a set of DNA sequences that has the right mismatch matrix, i.e. a set of sequences such that the distances in mismatches between any two are as required by the formal model of the system. It is not a priori obvious that sufficiently complex relationships can be designed. Moreover, to encode target machines, it is often advantageous to expanded the encoding to generate a larger mismatch matrix, but one whose requirements can be satisfied by real DNA sequences.

The sequential rules in our model act on complementary strands of DNA (3′-to-5′ for cycles one and three and 5′-to-3′ for cycle two), and we use this property to ratchet the computation forward. Thus the encoding we present will advance from step to step without going backwards. A programmed mutagenesis system can be made error-tolerant by increasing the length of its rules, permitting one or more error bases to be corrected on the next rewrite cycle.

§3 Results

We now turn to the practicality of the primitive operations of a programmed mutagenesis system. We have constructed the unary counter machine shown in Fig. 1, and have operated it through three cycles to gather efficiency data. We have previously demonstrated that the primitive operations required for programmed mutagenesis are functional.[16] That experiment used a system similar to the one explored here, but operated it only through the first two cycles of mutagenesis, and in the absence of the outside primer MLP.

The cycle reactions contained 0.02 uMolar of M1 and M2 oligonucleotides, ~0.2 uMolar of double stranded template DNA, Taq thermostable ligase (40 U) (NEB, #MO208), Vent thermostable polymerase (0.25 U) (NEB, #MO254), 0.2 uMolar of outer primers, and 1X ThermoPol buffer (NEB) supplemented with 1mM NAD. These reactions were thermal cycled for 1 minute at 94C, and 30 minutes at 45C for the indicated number of cycles.

Figure 3 shows a representative sample of our experimental data. To estimate the amount of products designated by II, III, and IV in Figure 1 we ^{32}P end-labeled M1 (for II and IV) and M2 (for III). Failed ligations produce products of characteristic lengths. The results of the reactions were run on polyacrylimide denaturing gels, and bands were quantiated on a phosphoroimager.

Because in cycle three the full-length band is a mixture of product IV and a shortened version of product II (as illustrated by the diagrams 7 and 6 in Fig. 3, respectively), it is impossible to directly ascertain that cycle three has occurred, or what the efficiency of that cycle is, from the full-length product. However, both the characteristic product of cycle three (diagram 8 in Figure 3) and the coloring of the characteristic product of cycle two (diagram 10 in Figure 3) are present, and indicate that cycle three has indeed taken place. We directly quantiated the amount of product in the characteristic band of cycle three, and estimated the amount of product IV present by assuming that the ligation efficiency of the third cycle is the same as that in the first cycle.

Because reactions proceed for the indicated number of cycles, while radiolabeled oligonucleotides are present in the reactions from the start, it is expected that the products of earlier cycles will accumulate as the reaction proceeds. Thus, it is expected and observed that characteristic bands of a particular cycle will be fainter than the bands which account for the product which has been accumulating in the reaction through the previous cycles.

We measured the amount of product in each band on the gel and calculated efficiencies of the latter cycles based on the amount of full-length product produced in the previous cycle. Because the unary counter operates serially through the cycles, we have to consider the amount of product II to be 100% of the template available for the second cycle, and the amount of product III to be 100% of the template available for the third cycle.

The results showed that 5.4% of template I is converted to the characteristic product of cycle one (diagram 2 in Fig. 3), and 0.5% is converted to product II (diagram 1 in Fig. 3) in cycle one, resulting in an 9% ligation efficiency for cycle one. In cycle two, 0.07% of the label is found in the product III band (diagram 3 in Fig. 3), which corresponds to 12.7% of product II created in cycle one. In cycle three, 0.05% of the label is found in the characteristic band of cycle three (diagram 8 in Fig. 3), which corresponds to 35% of product III created in cycle two. The latter number is calculated by assuming that every strand present in the characteristic band incorporated M1 rewrite rule at both the first and third cycle positions, as illustrated in diagram 8 in Fig. 3. If that is the case, than each strand is double labeled, and we assumed that for our calculations. However, it is possible that some of the strands in the band include M1 only in the third cycle position, and are single labeled. Thus, the estimate of 35% above is the lower bound. Assuming that the estimate is correct, and assuming that the ligation efficiency of the third cycle is the same as that of the first, we calculate that 3.5% of product III is converted to product IV (diagram 7 in Fig. 3) in the third cycle. In addition, no product III was generated in the cycle two negative control (NC) reaction in the absence of primer M1 and the presence of M2.

We have shown here that unary counter operates through three cycles of mutagenesis with increasing efficiency, but decreasing overall yield. The increase in efficiency in cycles two and three is expected. This is because in template I the coding region is embedded in a 3kb plasmid, and the oligonucleotides used in the system have to compete for binding spots with the long template annealing back

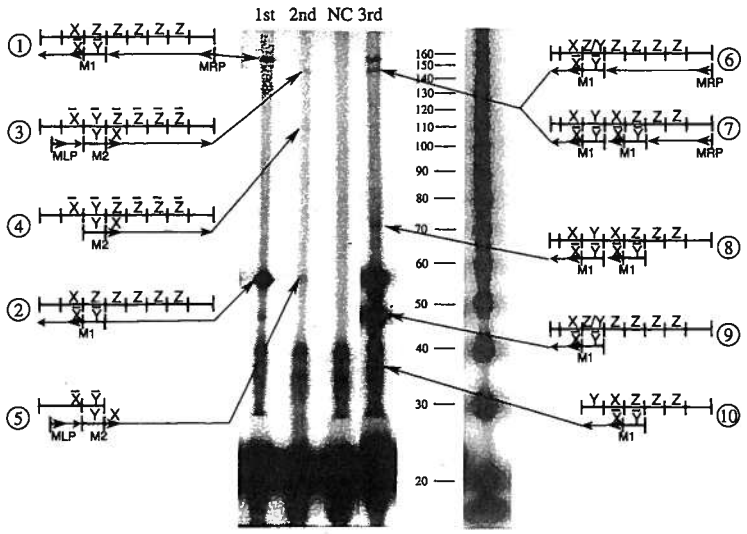

Fig. 3 Operation of the unary counter machine through cycles 1, 2, and 3, plus negative control (NC). Oligonucleotide M1 is end-labeled with ^{32}P in Cycles 1 and 3, while M2 is labeled in Cycle 2 and negative control. This is a polyacrylimide denaturing gel. Diagrams represent the products contained in the corresponding bands. Band labeled 3 is the full-length product of Cycle 2 (and the template for Cycle 3), designated by III in Fig. 1. Band labeled 8 is the characteristic length product of Cycle 3, produced when the product of extension of MRP fails to ligate to the product of extension of the mutagenic primer M1 annealed in the 3rd cycle location. The band of about 40 bp present in all lanes is the byproduct of oligonucleotide manufacturing. The band, of approximately double the size of the oligonucleotide, is present even when labeled oligonucleotides are run on a gel by themselves, with no template available. Other bands as illustrated.

on itself. Similar difficulties are not expected for the advanced cycles, resulting in the increased efficiency. However, the amount of product produced in the first cycle is such, that the yields of advanced cycles, even with the increased efficiency are low.

§4 Conclusion

We have shown that the basic operations of programmed mutagenesis, which is a universal model of computation, are functional, although the yield is low. The major impediment to continued cycling of the machine is that as long as template I is present, its products will increase exponentially with cycle number. However, if the Watson and Crick strands resulting from each DNA replication are separated into different compartments, then the compartment that receives product II will only contain a single computational state and thus will not repeat earlier computational steps. In-vivo programmed mutagenesis might be an effective way to computationally evolve DNA sequences and could potentially assist in sequence specific control of cellular function. It is also possible that gene conversion events and other natural systems for DNA evolution

could implement a more complex computational substrate than is now understood.

Acknowledgements

The authors thank Professors Douglas Melton, Maurice Fox and Leonard Lerman for the use of their lab space.

References

1) Khodor, J. and Gifford, D. K., "Programmed Mutagenesis is a Universal Model of Computation," to appear in a special issue of *Lecture Notes in Computer Science*, Springer-Verlag, 2002.

2) Khodor, J. and Gifford, D. K., "Programmed Mutagenesis is Universal," to appear in a special issue of *Theory of Computing Systems*, Springer-Verlag, 2002.

3) Adleman, L., "Molecular Computation of Solutions to Combinatorial Problems," *Science, 266, 5187*, pp. 1021–1024, 1994.

4) Lipton, R. J., "DNA Solution to Computational Problems," *Science, 268, 5210*, pp. 542–545, 1995.

5) Boneh, D., Dunworth, C., Lipton, R. J., and Sgall, S., "On the Computational Power of DNA," *Discrete Applied Mathematics, 71, 1–3*, pp. 79–94, 1996.

6) Komiya, K., Sakamoto, K., Gouzu, H., Yokoyama, S., Arita, M., Nishilkawa, A., and Hagiya, M., "Successive State Transitions with I/O Interface by Molecules," *in Proc. of Sixth International Meeting on DNA Based Computers, Preliminary*, pp. 21–30, 2000.

7) Sakamoto, K. Gouzu, H., Komiya, K., Kiga, D., Yokoyama, S., Yokomori, T., and Hagiya, M., "Molecular Computation by DNA Hairpin Formation," *Science, 288, 5469*, pp. 1223–1226, 2000.

8) Kari, L. and Thierrin, G., "Contextual Insertions/Deletions and Computability," *Information and Computation, 131*, pp. 47–61, 1996.

9) Landwebber, L. F. and Kari, L., "The Evolution of Cellular Computing: Nature's Solution to a Computational Problem," *Biosystems, 52*, pp. 3–13, 1999.

10) Beaver, D., "Molecular Computing," *1st DIMACS Workshop on DNA-based computers, Princeton, DIMACS Series, 27*, pp. 29–36, 1996.

11) Head, T. "Formal Language Theory and DNA: an Analysis of the Generative Capacity of Specific Recombinant Behaviors," *Bulletin of Mathematical Biology, 49*, pp. 737–759, 1987.

12) Paun, G., "Computing with Membranes," *J Comput Syst Sci, 61*, pp. 108–143, 2000.

13) Winfree, E., "On the Computational Power of DNA Annealing and Ligation," *DNA Based Computers II: DIMACS Workshop, June 10-12, 1996, American Mathematical Society*, pp. 191–213, 1998.

14) Mao, C., LaBean, T. H., Reif, J. H. and Seeman, N. C., "Logical Computation Using Algorithmic Self-assembly of DNA Triple-crossover Molecules," *Nature, 407*, pp. 493–496, 2000.

15) Hartemink, A. J., Gifford, D. K. and Khodor, J., "Automated Constraint-Governed Nucleotide Sequence Selection for DNA Computation," *Biosystems, 52*, pp. 93–97, 1999.

16) Khodor, J. and Gifford, D.K. "Design and Implementation of Computational Systems Based on Programmed Mutagenesis," *Biosystems, 52*, pp. 227–235, 1999.

17) *Current Protocols in Molecular Biology (Ausubel, I. and Frederick, M., eds), 8.5*, John Wiley & Sons, Inc, 1997.

Julia Khodor, Ph.D.: She has just received her Ph.D. in Electrical Engineering and Computer Science from MIT and is now enjoying her time off with her new daughter. She received her B.S. in Mathematics with Computer Science and B.S. in Biology in 1996 and her M.S. in Computer Science in 1998, all from MIT. Her graduate research was in the area of biological computing, primarily focusing on programmed mutagenesis. She is looking forward to the joys and challenges of an academic career.

David K. Gifford, Ph.D.: He is a professor of computer science and electrical engineering at MIT, where he leads a research group investigating issues in computational functional genomics. His research interests include understanding data from high-throughput experimental systems using probabilistic modeling techniques. He received a Ph.D. in computer science from Stanford University.